A PIC compatible RISC CPU core Implementation for FPGA based Configurable SOC platform for Embedded Applications

Haritha Madasi [1], S Nagakishore Bhavanam [2], Vaujadevi Midasala [3]

[1] Holy Mary Institute of Technology & Science, E.C.E Dept., Hyderabad, India
Email: harikha_bujji@yahoo.com
[2] Guru Nanak Institute of Science & Technology, E.C.E Dept., Hyderabad, India
Email: kishorereddy.vlsi@gmail.com
[3] Keshav Memorial Institute of Technology, E.C.E Dept., Hyderabad, India
Email: vasujadevi@gmail.com

***Abstract*—Modern embedded systems are built around the soft core processors implemented on FPGA. The FPGAs being capable of implementing custom hardware blocks giving the advantage of ASICs, and allowing the implementation of processor platform are resulting in powerful Configurable-system on chip(C-SoC)platforms. The Microchip's PIC microcontroller is very widely used microcontroller architecture across various embedded systems. The implementation of such core on FPGA is very much useful in CSOC based embedded systems. This type of designs can be widely used in those controlling fields demanding low power consumption and high ratio of performance to price. In this project a reduced instruction set computer (RISC) CPU IP core whose instructions are compatible with the Microchip PIC16C6Xseries of microcontrollers is implemented in VHDL. The core is based on 8-bit RISC architecture and top-Down design methodology is used in developing the core. The RISC CPU core is based on Harvard architecture with 14-bit instruction length and 8-bit data length and two-stage instruction pipeline. The architecture will be designed aiming at single cycle execution of the instructions, except those related to program branches. Since this type of CPU based on RISC architecture, there are only 35 reduced instructions in its instruction set, which are easy to be learned and used. The performance of the 8-bit RISC CPU is better than those of CPUs which are based on CISC architecture. Modelsim Xilinx Edition (MXE) will be used simulation and functional verification. The Xilinx Spartan-3E FPGAs will be used synthesis and timing analysis. The results will be verified on chip with chipscope tool.**

Keywords- CPU; RISC; IP Core; IC; MXE; Xilix.

I. PIC MICROCONTROLLERS

PIC microcontrollers are finding their way into new applications like smart phones, audio accessories, video gaming peripherals and advanced medical devices. Microchip provides solutions for the entire performance range of 8-bit microcontrollers, with easy-to-use development tools, complete technical documentation and post design-in support through a global sales and distribution network. There are hundreds of 8-bit PIC microcontrollers to choose from ranging from 6 to 100 pins and up to 128 KB Flash that are pin and code compatible across the portfolio. PIC microcontrollers with XLP technology feature the world's lowest active and sleep power consumption with flexible power modes and wake-up sources.MPLAB® Integrate development Environ-ment supports all PIC microcontrollers with C Compiler support and common development boards. Peripheral integration is key with communication and control peripherals like SPI, I2C™, UART, PWM, ADC and comparators as well as specialized peripherals for USB, mTouch™ Sensing, LCD, CAN and Ethernet. Customers have made PIC MCUs a worldwide standard, with over one million development systems shipped. PIC microcontrollers are quick and easy to design into a wide variety of applications with a long history of dependable product delivery[1].

II. INTRODUCTION TO PIC 16X MICROCONTROLLER

The PIC16C63A/65B/73B/74B devices are low-cost, high-performance, CMOS, fully-static, 8 bit microcontrollers in the PIC16CXX mid-range family.All PICmicro® microcontrollers employ an advanced RISC architecture[2]. The PIC16CXX microcontroller family has enhanced core features, eight-level deep stack and multiple internal and external interrupt sources. The separate instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with the separate 8-bit wide data. The two stage instruction pipeline allows all instructions to execute in a single cycle, except for program branches, which require two cycles. A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set gives some of the architectural innovations used to achieve a very high performance[1]. The PIC16C63A/73B devices have 22 I/O pins. The PIC16C65B/74B devices have 33 I/O pins. Each device has 192 bytes of RAM. In addition, several peripheral features are available including: three timer/ counters, two Capture/Compare/PWM modules and two serial ports. The Synchronous Serial Port (SSP) can be configured as either a 3-wire Serial Peripheral Interface (SPI) or the two-wire Inter-Integrated Circuit(I2C) bus. The Universal Synchronous Asynchronous Receiver Transmitter (USART) is also known as the Serial Communications Interface or SCI. Also, a 5- channel high-

DOI: 01.IJEPE.02.02. 44

speed 8-bit A/D is provided on the **PIC16C73B,** while the **PIC16C74B** offers 8 channels. The 8-bit resolution is ideally suited for applications requiring low-cost analog interface, e.g., thermostat control, pressure sensing, etc. The PIC16C63A/65B/73B/74B devices have special features to reduce external components, thus reducing cost, enhancing system reliability and reducing power consumption. There are four oscillator options, of which the single pin RC oscillator provides a low-cost solution, the LP oscillator minimizes power consumption, XT is a standard crystal, and the HS is for high speed crystals. The SLEEP (power-down) feature provides a power-saving mode. The user can wake up the chip from SLEEP through several external and internal interrupts and resets. A highly reliable Watchdog Timer (WDT), with its own on-chip RC oscillator, provides protection against soft- ware lock-up, and also provides one way of waking the device from SLEEP. A UV erasable CERDIP packaged version is ideal for code development, while the cost-effective One-Time-Programmable (OTP) version is suitable for production in any volume. The PIC16C63A/65B/73B/74B devices fit nicely in many applications ranging from security and remote sensors to appliance control and automotive[1]. The EPROM technology makes customization of application programs (transmitter codes, motor speeds, receiver frequencies, etc.) extremely fast and convenient. The small footprint packages make this microcontroller series perfect for all applications with space limitations. Low cost, low power, high performance, ease of use and I/O flexibility make the PIC16C63A/65B/73B/74B devices very versatile, even in areas where no microcontroller use has been considered before (e.g., timer functions, serial communication, capture and compare, PWM functions and coprocessor applications)[3].

III. RISC IP DESIGN

The CPU IP Core is an instruction-compatible of the Microchip PIC16C6X -series microcontrollers. The CPU having 22bit I/O ports only has 35 instructions which make the final hardware implementation compact. The CPU can address up to 2048 instruction words. And we use two level pipe-line structures to improve the executing speed. Figure 3-1 below shows the main modules of the RISC CPU IP Core[2].

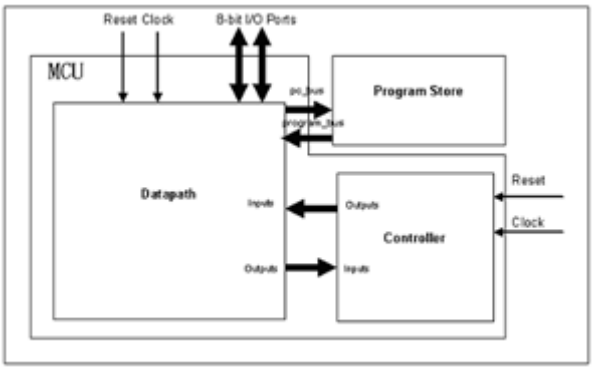

Figure 3-1. Main modules of RISC CPU IP Core

As figure 3-1 show, the CPU module contains the datapath and controller modules while the program store is external to the CPU Core. This means that the Program Store can be located off-chip if needed. Locating the Program Store on-chip however enables the CPU Core to execute at the internal clock speed of CPU.

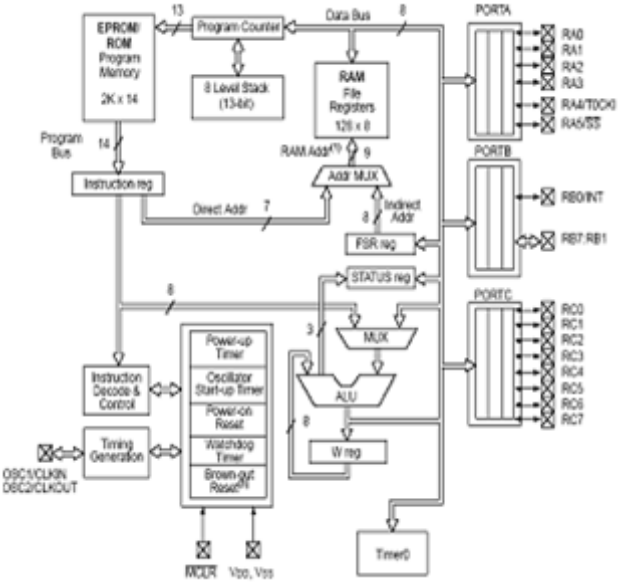

Figure 3-2. CPU architecture

Figure 3-2 shows the CPU architectural overview. To begin with, the CPU uses Harvard architecture, in which, program and data are accessed from separate memories using separate buses. This improves bandwidth over traditional von eumann architecture where program and data may be fetched from the same memory using the same bus. Separating program and data busses further allows instructions to be sized differently than 8-bit wide data words. Instruction words are 14-bits wide making it possible to have all single word instructions. A 14-bit wide program memory access bus fetches a 14-bit instruction in a single cycle. A two-stage pipeline overlaps fetch and execution of instructions. Consequently, all instructions execute in a single cycle except for program branches. The CPU can directly or indirectly address its register files or data memory. All special function registers (including the program counter) are mapped in the data memory. The CPU has a symmetrical instruction set that makes it possible to carry out any operation on any register using any addressing mode. This symmetrical nature makes programming with the CPU simple yet efficient, thus significantly reducing the learning curve. The CPU contains an 8-bit ALU and working register (W). The ALU is a general arithmetic unit. It performs arithmetic and Boolean functions between data in the working register and any register file. The ALU is 8-bits wide and capable of addition, subtraction, shift, and logical operations. Unless otherwise mentioned, arithmetic operations are two's complement in nature. In two-operand instructions, typically one operand is the working register (W register), the other operand is a file register or an immediate constant. In single operand instructions, the operand is either the working register or a file register.

DOI: 01.IJEPE.02.02. 44

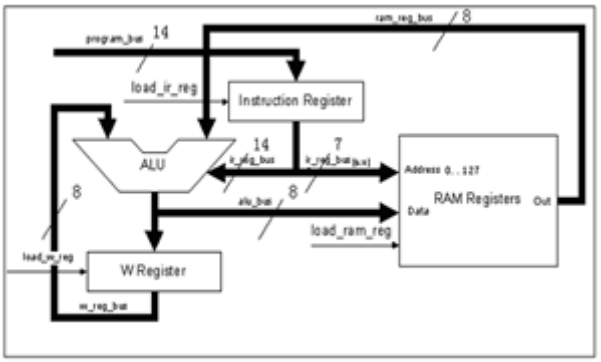

Figure 3-3. Simplified datapath

Figure 3-3 shows a somewhat simplified datapath architecture where the eight Special Function Registers are not shown. This datapath is sufficient to execute most instructions so it will be used to introduce the overall CPU IP Core architecture.Most instructions in the datapath architecture operate on either the 'W' or one of the 'f' registers. The 'W' register is input as the ALU's first operand while the currently addressed 'f' register is input as the second ALU operand. The ALU result is output on 'alu_bus' from where it is loaded into either the 'W' register or into the currently addressed 'f' register. The Controller either asserts the 'load_w_reg' or the 'load_ram_reg' depending on if the instruction word in the instruction Register directs the result to the 'W' or 'f' register respectively [4]. In order for the ALU to be able to update the STATUS register bits Z, C, and DC a separate 'status_bus' exists together with three control signals named 'load_z', 'load_d' and 'load_dc'. These signals are displayed below in figure 4. Also note that figure 4 contains the 'ALU Mux' on operand input two which allows the Controller to select the second ALU operand to come from either the RAM Registers (shown above in figure 3) or from one of the eight SFR (Special Function Register) Registers [0..7]. The ALU knows from the 'ir_reg_bus' which instruction word is currently being executed. The 'carry_bit' input is connected to the C bit in the STATUS register. This is used during additions and subtractions. The 'ram_data_bus' input to the ALU Mux comes from the RAM registers as shown above in figure 3-4. The 'sfr_data_bus' comes from the currently addressed SFR register.

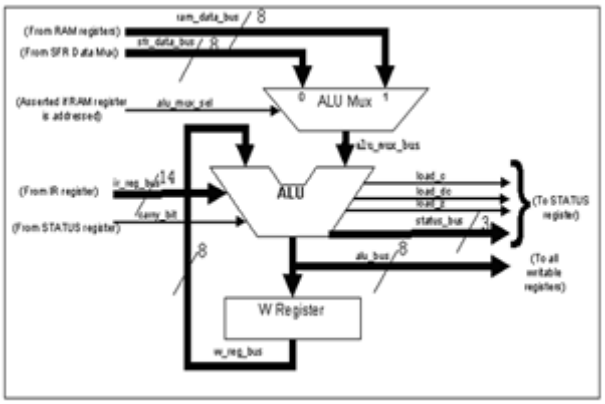

Figure 3-4. Datapath surrounding ALU

The CPU architecture allows both direct and indirect addresses. A direct address is encoded directly in the 14-bit instruction word while an indirect address is taken from the FSR Special Function Register. The address generation logic is shown below in figure3-5.

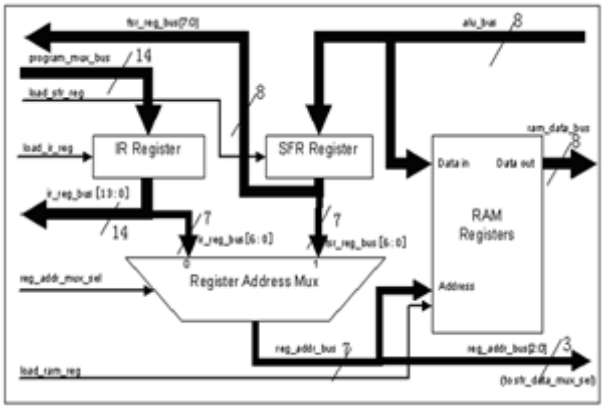

Figure 3-5. Direct and indirect address selection

As can be seen in Figure 3-5, the Register Address Multiplexer passes through either the 'fffffff' field in the instruction word (bits 6:0) or the value in the SFR register. The 'reg_addr_mux_sel' signal determines which source bus to route to 'reg_addr_bus'. The CPU architecture has an eight-level stack that enables subroutines to be called via the CALL/RETLW instruction pair. Jumps to an absolute address can also be done via the GOTO instruction. Figure 3-6 below shows the Program Counter Datapath required to allow the CALL, RETLW and GOTO instructions to be implemented.

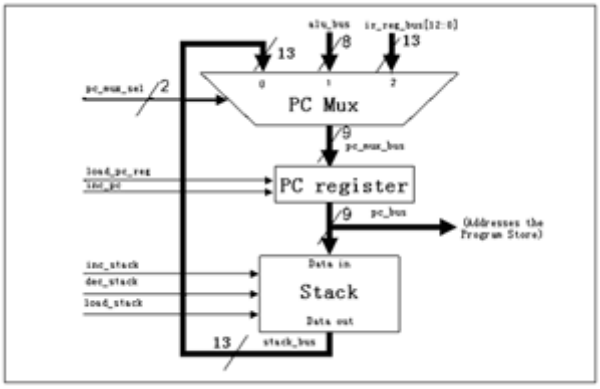

Figure 3-6. Program counter and stack

As Figure 3-6 shows, the 'pc_mux_sel' input selects the source of the data on the 'pc_mux_bus'. The 'pc_ register' register is loaded on the rising edge of the clock signal when the 'load_pc_reg' signal is asserted. The 'inc_pc' signal is used to increment the Program Counter as each instruction is executed. The 'inc_stack' and 'dec_stack' signals are used to select which stack register will be loaded when the 'load_stack' signal is asserted. The 'stack_bus' always outputs the data in the current stack location (i.e. the data previously written to the stack location)[5]. The CPU IP Core contains two bidirectional 8-bit I/O and a bidirectional 6-bit I/O ports .The I/O port bits can individually be programmed as input or output via the TRIS instruction. The output data (for output ports) is programmed into GPIO registers. Figure 3-7 below shows the internal architecture of the 'tri_state_port' module.

DOI: 01.IJEPE.02.02. 44

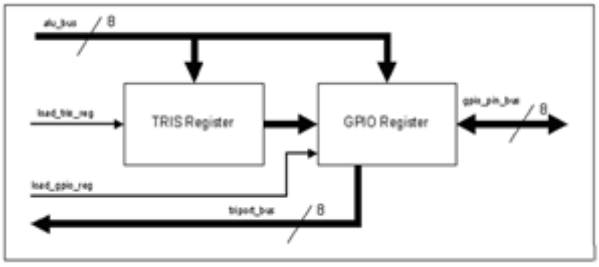

Figure 3-7. I/O port logic

As can be seen in figure 3-7, the 'alu_bus' input allows the TRIS and GPIO 8-bit registers to be loaded from 'alu_bus'. A TRIS registers is loaded from 'alu_bus' when a TRIS instruction is executed. A GPIO register is loaded when Special Function Registers is the write target of an instruction [7] (i.e. MOVWF). Setting a bit in the TRIS register puts the corresponding output driver in a hi-impedance mode. Clearing a bit in the TRIS register puts the contents of the output latch on the selected pin.

Controller Architecture:

As shown in Figure 3-1, the controller module is connected via a number of inputs and outputs to the 'datapath' module. The controller is responsible for decoding the current instruction in the IR (instruction register) and set up the datapath control signals so that the data is correctly routed from the source to the destination register accordingly to the semantics of the instruction. As the instruction in the IR changes so does the functionality of the Controller. The Controller is essentially a simple state machine that asserts a pre-determined number of signals for the various instructions. Figure 3-8 below shows the inputs and outputs of the 'controller' module.

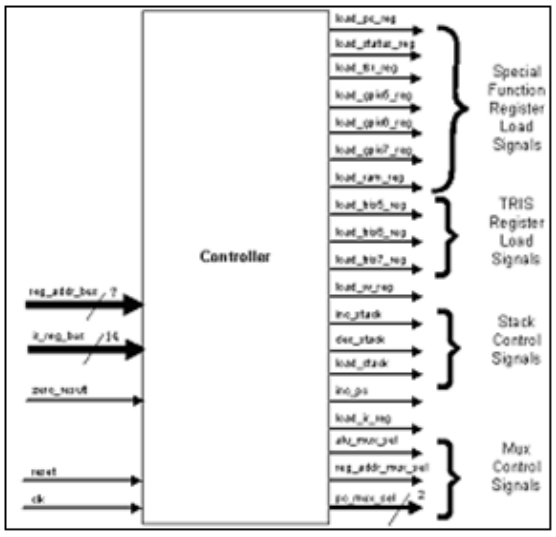

Figure 3-8. Controller

IV. SIMULATION RESULTS

Figure 4-1 shows the top level simulation waveforms. This figure shows RAM, ALU and instruction decoder waveforms also for comparison. These waveforms are explained separately in subsequent figures.

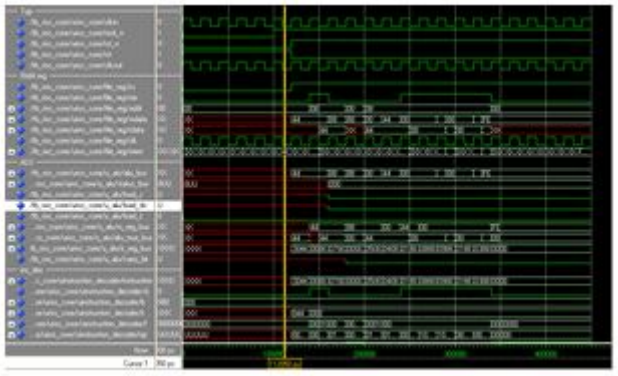

Figure 4-1. Simulation waveforms

Figure 4-2 shows instruction decoder waveforms. Instruction decoding happens based on the instruction format described in Chapter 2. MSB 2-bits indicate whether the instruction is byte oriented or bit oriented or control instruction. For the byte oriented instruction shown in Figure 4-2, bits 6 down to 0 is the register address for read/write.

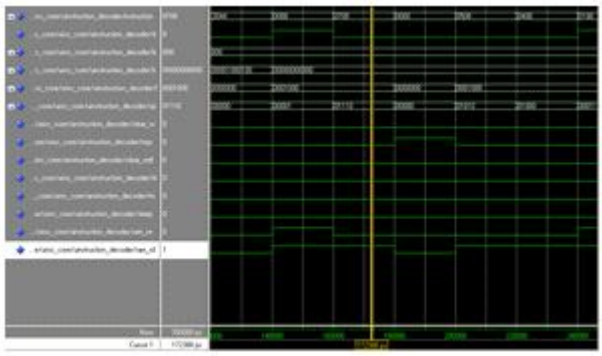

Figure 4-2. Instruction decoder waveforms

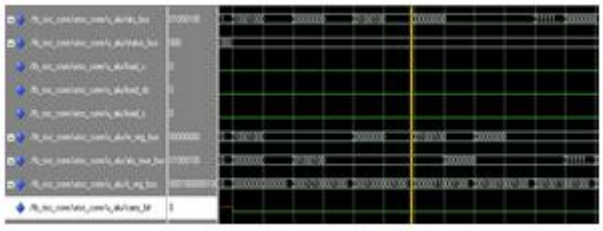

Figure 4-3. ALU waveforms

As shown in Figure 4-3, alu_mux_bus and w_reg_bus are the inputs to the ALU. The output comes out of alu_bus. The operation to be performed is decided by the instruction coming on ir_reg_bus. The status flags are shown on load_c, load_dc and load_z signals.

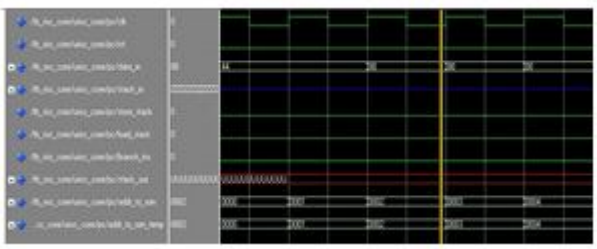

Figure 4-4. Program counter waveforms

As shown in Figure 4-4, the output of program counter comes on addr_to_rom. This is changed based on the type of instruction executed. By default, it is incremented by '1'.

DOI: 01.IJEPE.02.02. 44

In case of branch instructions (indicated by branch_ins signal), the PC is incremented by value denoted on data_in. Similarly, the current address is saved to stack through stack_out (push instruction).

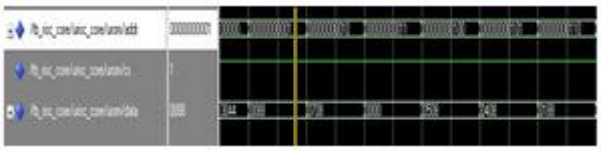

Figure 4-5. ROM waveforms

Figure 4-5 shows ROM waveforms where the address, chip select and corresponding instruction can be seen on data.

V. FPGA IMPLEMENTATION

The design is implemented in Spartan 3E FPGA XC3S500E on Spartan 3E starter kit. The design occupies 3% of the FPGA and correspondingly maximum frequency of operation is 233.91MHz. The placed design is shown in Figure 5-1. Routed design is shown in Figure 5-2.

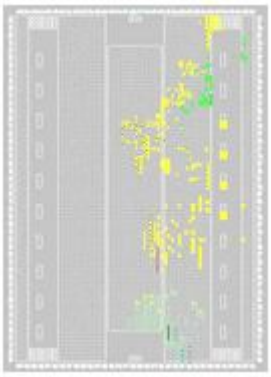

Figure 5-1. Placed design

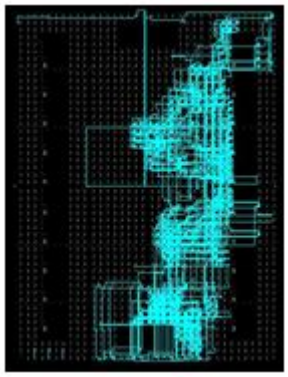

Figure 5-2. Routed design

The design is implemented in Spartan 3E FPGA and the results are verified using chipscope on-chip debugging tool. The results obtained are shown in Figure 5-3.

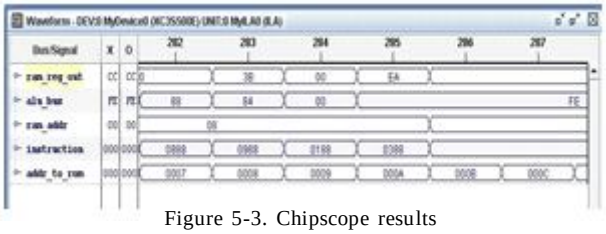

Figure 5-3. Chipscope results

For on-chip debugging, some important signals like ROM address, instruction, register address, register data input and data output are probed. Based on the instruction being executed, we can see the signal values. The results are verified using chipscope analyzer tool from Xilinx.

ACKNOWLEDGEMENT

We would like to thank the Faculty of GNI, KMIT & HMIT for their support. We acknowledge Mr satish Babu help in the development and testing of our implementation "A PIC compatible RISC CPU core Implementation for FPGA based Configurable SOC platform for Embedded Applications" .

CONCLUSION

We are Implemented RISC based CPU Architecture. This CPU uses hardware Architecture and two level Pipelined Structures. This Architecture have only 35 reduced instruction in it's instruction set and most of the instructions only need one machine cycle to be executed. The performance of the CPU has been improved by replacing micro program with direct Logic Block.This CPU gives good performance ratherthan SISC based Architectures.the Simulation Results shows the CPU has a good performance.

REFERENCES

[1] PIC16C63A/65B/73B/74B DataSheet, Microchip Technology Inc,2000.

[2] Piguet C. Low power design of 8-b embedded cool-RISC microcontroller cores. IEEE Journal of Solid-State Circuits, 1997, 32(7): 1067~1077

[3] OTP 8-Bit CMOS MCU with EEPROM Data Memory, USA: Microchip Technology Inc, 1998

[4] PIC micro TM Mid-Range MCU Family Reference Manual. Microchip Technology Inc.1997. 1~688.

[5] Mano, M. M., Computer System Archetecture, Prentice Hall, Tsinghua University press,1997.

[6] Ma G K, Taylor F J. Multiplier policies for digital signal processing. IEEE ASSP Magazine, 1990, 7(1): 6~20

[7] Hewlett-Packard Company (September 1987). *Hewlett-Packard Journal* **38**

[8] Wayne Wolf, Modern VLSI Design, USA: Printice Hall, 2001:298~299.

About Authors

S Nagakishore Bhavanam is an M.Tech Post-Graduate of VLSI-SD from Aurora's Technological & Research Institute (ATRI), Department of ECE, JNTUH. He obtained his B.Tech from S.V.V.S.N Engineering college, Ongole. He has 18 Months of teaching experience. He has **4** Research Papers, Published in IEEE Xplore, He has **3** International journal Publications and has **6** Papers, Published in International & National Conferences. His interesting Fields are Low Power VLSI, Digital System Design, Sensor Networks, and Communications.

Vasujadevi Midasala is an M.Tech Post-Graduate of VLSI from Bandari Institute of science and Technology, Department of ECE, JNTUH. She obtained her B.Tech from S.V.V.S.N Engineering college Ongole. She Has 4 Research Papers Published in International/ National Journals. Her interesting Fields are Low Power VLSI, Design for Testability.

Haritha M is an M.Tech Post- Graduate of VLSI from, Aurora's Technological & Research Institute (ATRI), Department of ECE, JNTUH. She obtained her B.Tech from T.K.R Engineering college Hyderabad. Her interesting Fields are Microcontrollers & Signals and Signals.

DOI: 01.IJEPE.02.02. 44